NASA Contractor Report 177575

2681
p23

Cascading a Systolic Array and a Feedforward Neural Network for Navigation and Obstacle Avoidance Using Potential Fields

Edward S. Plumer

CONTRACT NGT-50642
February 1991

NASA
National Aeronautics and Space Administration

NASA Contractor Report 177575

Cascading a Systolic Array and a Feedforward Neural Network for Navigation and Obstacle Avoidance Using Potential Fields

Edward S. Plumer

Department of Electrical Engineering, Stanford University
Stanford, California

Prepared for
Ames Research Center
CONTRACT NGT-50642
February 1991

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

1 Abstract

The goal of this research is the development of a technique for vehicle navigation and control in the presence of obstacles. A potential function has been devised that peaks at the surface of obstacles and has its minimum at the proper vehicle destination. This function is computed using a systolic array and is guaranteed not to have local minima. A feedforward neural network is then used to control the steering of the vehicle using local potential field information. In this case, the vehicle is a trailer-truck backing up. Previous work has demonstrated the capability of a neural network to control steering of such a trailer-truck backing to a loading platform, but without obstacles. Now, the neural network has been able to learn to navigate a trailer-truck around obstacles while backing toward its destination. The network is trained in an obstacle free space to follow the negative gradient of the field, after which the network is able to control and navigate the truck to its target destination in a space of obstacles which may be stationary or movable.

2 Introduction

The methods of backpropagation and backpropagation-through-time [1,2,3] are useful in many neural-control applications [4,5,6,7,8]. Nguyen and Widrow have successfully used the technique to train a multi-layer neural network to act as a nonlinear state feedback controller for backing a trailer-truck to a loading dock [6,7]. This paper examines a computationally efficient method of modifying the work of Nguyen and Widrow to permit truck backing in the presence of obstacles.

Robotic path-planning and obstacle avoidance has been approached successfully using a variety of techniques [9,10,11,12]. For example, Khatib has proposed a method of obstacle avoidance for robotic systems using artificial potential fields defined in "operational space"[1]. Using this method, a potential field in the vicinity of obstacles induces repulsive forces on the robot [10]. When combined with an acceleration term toward the goal and the robot's dynamic equations, it is possible to calculate analytically the desired trajectory and control forces. Variants of this formulation have been used by others as well [13,14,15,16].

The analytical nature of the artificial potential fields used by Khatib does not lend itself directly to a neural network implementation. However, the concept of low-level control and navigation based on such potential fields is quite useful and motivates the technique for truck-backing presented in this paper. The backing trailer-truck is the vehicle model used in this study for the development of learning systems which are capable of navigation in the presence of obstacles, nevertheless, the method is applicable to other vehicles.

In this paper, a hybrid network technique for truck navigation based on potential fields is presented. The technique involves the calculation of a potential field using a systolic array as shown in section 3. Several algorithms for the systolic array are presented and the properties of the resulting fields are analyzed. This is followed, in section 4, by a discussion of how a multi-layer feedforward neural network can be trained to use local potential field information to control truck steering.

3 Potential Field for Navigation

The role of the potential field is to distribute information about obstacle and goal locations throughout operational space, thus providing local information about how to solve the global navigation problem. A field is developed which is a measure of the *penalty* associated with a given location in obstacle space. A secondary controller can then use that local information to navigate by attempting to follow the negative field gradient. This field directs the truck toward the goal and away from obstacles. In light of its role, the potential field should possess certain properties:

1. The field should decrease as the goal is approached so that following the negative gradient will drive the truck toward the goal.

[1] *operational space:* space in which avoidance task is defined as opposed to a transformed *joint space* [10]

2. The field should increase rapidly in the vicinity of an obstacle to avoid collisions.

3. The field should have a single local minimum (the goal) to prevent the truck from getting trapped at some undesired location.

3.1 Notation

3.1.1 Operational Space Representation

Obstacle representation directly affects the choice of algorithms used to solve the obstacle avoidance problem. In previous works, obstacles have been represented in many ways, including polygons [11] and sets of known object shapes (squares, circles etc.) placed at specified coordinates [10].

In this work, navigation takes place within a 2 dimensional operational space. The space is partitioned by a rectangular lattice with grid spacing δ into grid-squares whose centers are at lattice points. A particular lattice point is denoted by $n_{i,j} = (i\delta, j\delta)$ and the grid-square that contains it is denoted by $g_{i,j}$. With these definitions we can specify obstacles as sets of such grid-squares which corresponds to regions of operational space where the truck cannot navigate. The set of lattice points associated with obstacles is denoted by $\mathcal{O}$. The target destination for the truck is called the goal and is denoted by n_G. It is assumed that the goal is located at a lattice point and is not contained within an obstacle. This representation, and the proposed algorithms, presumes that obstacle locations within operational space are known, possibly via an over-head camera. An example obstacle space, which will be used throughout this paper, is shown in figure 1. The boundary of the operational space will also be considered as an obstacle or barrier.

A notion of lattice point neighbors is needed to discuss the potential field algorithms.

Definition 1 (Neighbor) Two lattice points are said to be neighbors if they are separated by distance δ. The set of 4 neighbors for a lattice point, $n_{i,j}$, will be denoted as

$$N_4(n_{i,j}) = \{n_{i+1,j}, n_{i-1,j}, n_{i,j+1}, n_{i,j-1}\}.$$

Two lattice points will be called extended neighbors if they are separated by distance δ or $\delta\sqrt{2}$. The set of 8 extended neighbors for a lattice point, $n_{i,j}$, will be denoted as

$$N_8(n_{i,j}) = \{N_4(n_{i,j}), n_{i+1,j+1}, n_{i-1,j-1}, n_{i+1,j-1}, n_{i-1,j+1}\}.$$

A concept of "connectedness" for lattice points will also be useful. This definition is similar to the notion of "path connected" when speaking of a subset of $\Re^n$ and describes whether there exists a path between two points which does not cross an obstacle. Defined formally we have:

Definition 2 (Lattice path-connected) Lattice points n_a and n_b will be called lattice path-connected if there exists an ordered set of lattice points, $p_{ab} = \{n_1, \cdots, n_N\} \notin \mathcal{O}$, called the path, such that $n_1 = n_a$, $n_N = n_b$, and $n_{i+1} \in N_4(n_i)$. A similar concept of extended-lattice path-connected can be defined using $N_8()$ instead of $N_4()$.

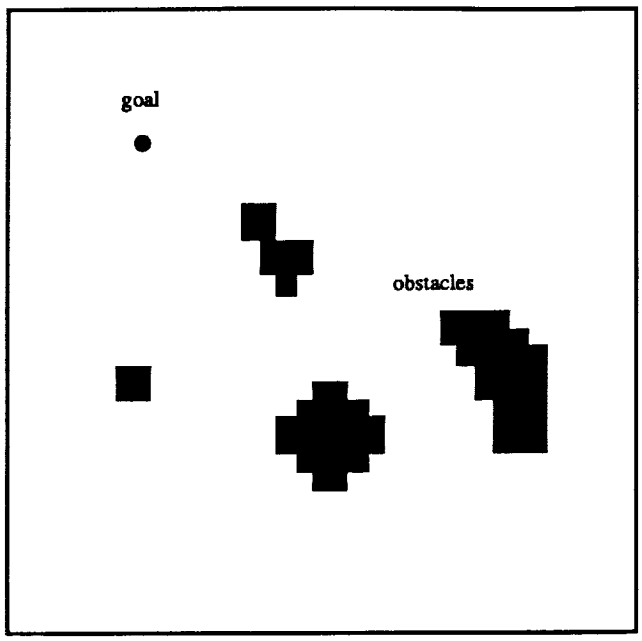

Figure 1: *A sample obstacle space. Obstacles are represented by black squares in a rectangular grid.*

3.1.2 Systolic Array for Computing Potential Fields

The regular representation of obstacles on a grid as presented above permits computation of the potential field by a set of simple, identical processors arranged in a rectangular array (one processor associated with each lattice point, $n_{i,j}$). Each processor senses whether the corresponding lattice point is contained in an obstacle, that is whether $n_{i,j} \in \mathcal{O}$. The processor then computes the local level of the potential field by interacting dynamically with neighboring processors. Such a structure is a type of systolic array. We introduce the following definitions:

Definition 3 (Lattice potential field) The potential field, denoted by $\Phi_{x,y}(k)$, is a 2-D function giving the value of the potential field at time iteration k at location (x, y). The set of values of the potential field at the lattice points $n_{i,j}$ at time iteration k define a lattice potential field, $\Phi_{i,j}(k)$.

Definition 4 (Lattice local minimum) A lattice point, n_a, will be called a lattice local minimum of $\Phi_{i,j}$ if $\Phi_{n_a} \leq \Phi_{n_b}\ \forall n_b \in N_4(n_a)$. Likewise, n_a, is said to be an extended-lattice local minimum of $\Phi_{i,j}$ if $\Phi_{n_a} \leq \Phi_{n_b}\ \forall n_b \in N_8(n_a)$. This conveys the notion of whether or not there is a neighboring point which is "downhill" from the current point.

The various methods presented for calculating the potential field involve relaxation of the interconnected processors while subjecting them to obstacle and goal positions. The specific relaxation equation implemented by all of the processors determines the properties of the potential field. We

develop a set of such equations in this paper, and examine the properties of the resulting potential fields.

3.2 Single Layer Relaxation Methods

In this section two relaxation methods are presented which seem most intuitive but which do not produce desirable potential fields. These methods are presented as motivation for the slightly more complicated method presented in section 3.3.

3.2.1 Local Averaging Relaxation

An obvious relaxation implementation is to fix the outputs of the goal-processor and obstacle-processors at two different levels while allowing the other processor outputs to converge to intermediate values by averaging the outputs of their nearest neighbors. The equations governing this process are:

$$\begin{aligned}
\Phi_{i,j}(0) &= 0 \\
\Phi_{i,j}(k+1) &= \tfrac{1}{4}\left(\Phi_{i+1,j}(k) + \Phi_{i-1,j}(k) + \Phi_{i,j+1}(k) + \Phi_{i,j-1}(k)\right) \quad \text{for} \quad \begin{cases} n_{i,j} \notin \mathcal{O} \\ n_{i,j} \neq n_G \end{cases} \\
\Phi_{i,j}(k+1) &= \Phi_{max} \quad \text{for} \quad n_{i,j} \in \mathcal{O} \\
\Phi_{i,j}(k+1) &= 0 \quad \text{for} \quad n_{i,j} = n_G
\end{aligned} \tag{1}$$

This results in a field which decreases smoothly toward the goal, and increases toward obstacles. Convergence of the relaxation equations is easily shown by noting that $\Phi_{i,j}(k)$ forms a bounded, non-decreasing sequence in k for each $n_{i,j}$. However, a unique local minimum, as defined above, is not guaranteed. It is possible, albeit rare, to have a point, $n_a \neq n_G$, where $\Phi_{n_a} = \Phi_{n_b}\ \forall n_b \in N_4(n_a)$. This approach also has more serious problems.

First, as can be seen in the example of figure 2, the field increases approximately logarithmically away from the goal. This can be derived by rearranging the relaxation equation

$$4\Phi_{i,j} = (\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1}) \tag{2}$$

into two separate terms

$$0 = (\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}) + (\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}) \tag{3}$$

Noting that each term approximates a second-derivative, this equation is then converted into a continuous model,

$$\frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = 0 \tag{4}$$

If we assume circular symmetry of the solution, the equation can be rewritten in terms of radius, r:

$$\frac{\partial^2\Phi}{\partial r^2} + \frac{1}{r}\frac{\partial\Phi}{\partial r} = 0 \tag{5}$$

The solution to this Euler equation is

$$\Phi(r) = C_1 + C_2 \ln r \tag{6}$$

The reader may note that the local-averaging calculation is equivalent to the numerical solution of Laplace's equation. The logarithmic growth rate implies that the resolution required in estimating the gradient of the field grows linearly with the distance from the goal. Using the feedforward neural-controller described in section 4, a linear growth rate of the field, requiring a constant level of accuracy for gradient estimation, is more desirable. Second, there is no parameter which controls the rate of growth. Hence, the resolution of the obstacle grid cannot be changed without affecting the cross-sectional shape of the potential field, nor can the radius of influence of obstacles be adjusted. Finally, for regions of operational space partially shielded from the goal by obstacles, the field is effectively "pinned" to the upper value of the potential field.

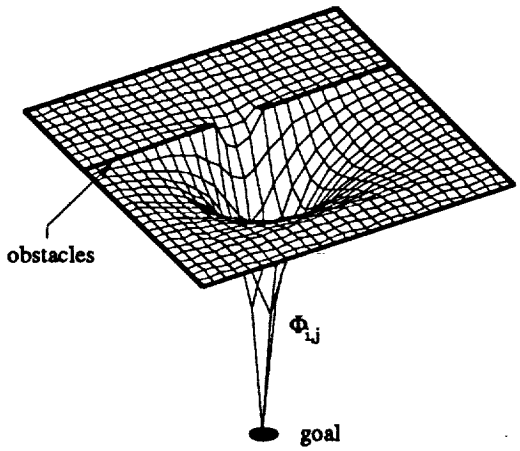

Figure 2: *Local-averaging relaxation solution showing the undesirable logarithmic growth rate.*

3.2.2 Distance-to-Goal Relaxation

In an attempt to correct the undesirable growth rate, a different technique is considered. This technique involves the calculation, for each field point, of the minimum distance to the goal along lattice paths not intersecting obstacles. The distance measure used here is a "city block metric." The relaxation equations are:

$$\begin{aligned}
\Phi_{i,j}(0) &= 0 \\
\Phi_{i,j}(k+1) &= 1 + \min \left\{ \begin{array}{l} \Phi_{i+1,j}(k), \\ \Phi_{i-1,j}(k), \\ \Phi_{i,j+1}(k), \\ \Phi_{i,j-1}(k) \end{array} \right\} \quad \text{for} \quad \left\{ \begin{array}{l} n_{i,j} \notin \mathcal{O} \\ n_{i,j} \neq n_G \end{array} \right. \\
\Phi_{i,j}(k+1) &\approx \infty \qquad \text{for} \quad n_{i,j} \in \mathcal{O} \\
\Phi_{i,j}(k+1) &= 0 \qquad \text{for} \quad n_{i,j} = n_G
\end{aligned} \tag{7}$$

This technique produces the desired linear growth rate. Furthermore, convergence of this method to a solution with unique local minimum is guaranteed. (the proofs are similar to those in lemmas 2 and 3). This method, however, does not provide a mechanism for repelling the truck away from obstacles. There is no penalty for coming close to an obstacle, only a mechanism for preventing a path from crossing through an obstacle.

3.3 A Two Layer Relaxation Method

A method which combines the obstacle repulsion property of the local-averaging technique with the linear growth-rate and unique local minimum properties of the distance-to-goal technique is presented next. The method relies on a 2-layered approach. A first processing layer uses the obstacle positions to compute a barrier-potential field, $\Phi^B_{i,j}$. A second processing layer uses the barrier-potential field and the goal position to compute an overall potential field, $\Phi^D_{i,j}$.

3.3.1 Barrier-Potential Field

The barrier-potential field, $\Phi^B_{i,j}$, is designed only to spread information about obstacle locations to surrounding nodes and therefore ignores the location of the goal. The relaxation equations are:

$$
\begin{aligned}
\Phi^B_{i,j}(0) &= 0 \\
\Phi^B_{i,j}(k+1) &= \max \left\{ \begin{array}{ll}
\alpha^{\sqrt{2}} & \Phi^B_{i+1,j-1}(k), \\
\alpha^{\sqrt{2}} & \Phi^B_{i-1,j-1}(k), \\
\alpha^{\sqrt{2}} & \Phi^B_{i+1,j+1}(k), \\
\alpha^{\sqrt{2}} & \Phi^B_{i-1,j+1}(k), \\
\alpha & \Phi^B_{i+1,j}(k), \\
\alpha & \Phi^B_{i-1,j}(k), \\
\alpha & \Phi^B_{i,j+1}(k), \\
\alpha & \Phi^B_{i,j-1}(k)
\end{array} \right\} \quad \text{for} \quad n_{i,j} \notin \mathcal{O} \\
\Phi^B_{i,j}(k+1) &= \Phi^B_{max} \quad \text{for} \quad n_{i,j} \in \mathcal{O}
\end{aligned}
\tag{8}
$$

The result of applying the calculation to the obstacle pattern presented earlier is shown in figure 3. The barrier potential field, $\Phi^B_{i,j}$, decays geometrically away from obstacles. The geometric ratio, α, is used to control the effective radius of influence of obstacles.

3.3.2 Overall Potential Field

The overall potential field, $\Phi^D_{i,j}$, measures a "generalized distance" from each lattice point to the goal. The generalized distance consists of the weighted sum of two terms. The first is an "extended city block distance" from the point to the goal along the "best" path. This is similar to a "city block metric," except that paths are allowed in 8 directions instead of the usual 4. The second term measures the cumulative barrier potential, $\Phi^B_{i,j}$, along the path. Thus, the potential field, $\Phi^B_{i,j}$, has

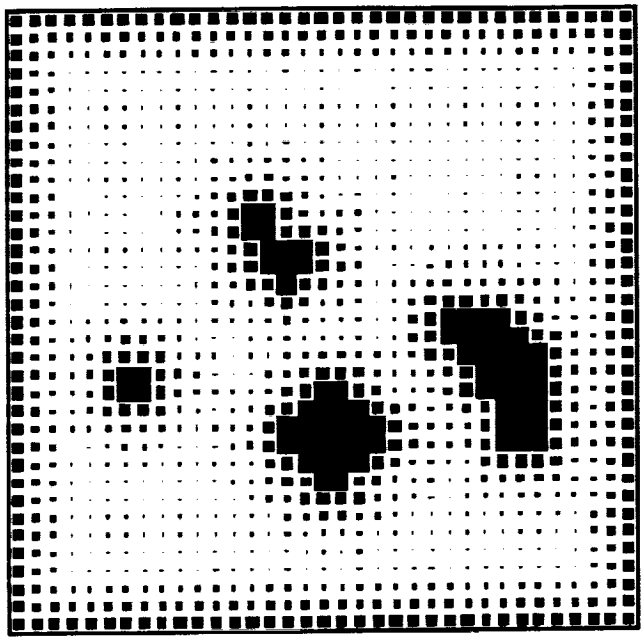

Figure 3: *Barrier potential field for the example obstacle pattern ($\alpha = 0.6$). Square sizes represent the values of $\Phi^B_{i,j}$.*

the effect of stretching out the distance metric in the vicinity of obstacles, hence favoring paths which steer clear of obstacles. The relative weighting of city-block distance vs. barrier-penalty is controlled using the parameter Φ^B_{max}. The "best" path in question is the path which minimizes this generalized distance. The neural-controller uses this overall field to perform the navigational task as will be shown in section 4. Once $\Phi^B_{i,j}$ has reached steady-state, the relaxation calculations for $\Phi^D_{i,j}$ are:

$$
\begin{aligned}
\Phi^D_{i,j}(0) &= \Phi^D_{max} \approx \infty \\
\Phi^D_{i,j}(k+1) &= \Phi^B_{i,j} + \min \left\{ \begin{array}{lll} \sqrt{2} + \Phi^D_{i+1,j-1}(k), \\ \sqrt{2} + \Phi^D_{i-1,j-1}(k), \\ \sqrt{2} + \Phi^D_{i+1,j+1}(k), \\ \sqrt{2} + \Phi^D_{i-1,j+1}(k), \\ 1 + \Phi^D_{i+1,j}(k), \\ 1 + \Phi^D_{i-1,j}(k), \\ 1 + \Phi^D_{i,j+1}(k), \\ 1 + \Phi^D_{i,j-1}(k) \end{array} \right\} \quad \text{for} \quad \left\{ \begin{array}{l} n_{i,j} \notin \mathcal{O} \\ n_{i,j} \neq n_G \end{array} \right. \\
\Phi^D_{i,j}(k+1) &= \Phi^D_{max} \approx \infty \quad \text{for} \quad n_{i,j} \in \mathcal{O} \\
\Phi^D_{i,j}(k+1) &= 0 \quad \text{for} \quad n_{i,j} = n_G
\end{aligned}
\tag{9}
$$

After $\Phi^D_{i,j}$ has converged, the minimum-distance path from a random initial lattice point can be found by repeatedly moving to the nearest neighbor with smallest $\Phi^D_{i,j}$. Setting Φ^D_{max} sufficiently

large prevents this minimum-distance path from ever "punching through" an obstacle rather than going around it, no matter the length of the path. The values of $\Phi^D_{i,j}$ calculated for the example obstacle pattern are shown in figure 4.

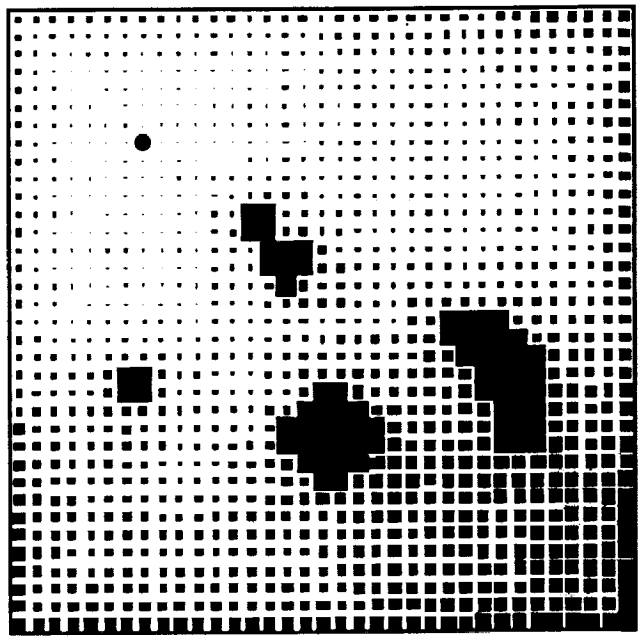

Figure 4: *Overall potential field ($\Phi^B_{max} = 50$, $\alpha = 0.6$). Square sizes indicate the level of $\Phi^D_{i,j}$.*

3.3.3 Features of Two Layer Relaxation Method

The two-layer relaxation method and the resulting lattice potential field have several desirable properties which will be explained in this section.

Finite Convergence Time — The relaxation process consists of a set of iterated equations. In lemmas 1 and 2, we show that the iterative process converges in a finite number of steps to a unique set of values. The convergence time of $\Phi^B_{i,j}$ is proportional to the maximum separation of obstacles and the convergence time of $\Phi^D_{i,j}$ is proportional to the length of the longest minimum-distance path to the goal.

Lemma 1 *The barrier potential field, computed by equation 8, converges in a finite number of steps.*

Proof. Consider any point not in an obstacle, $n_a \notin \mathcal{O}$. There exists a minimum length path, $p_{O_a} = \{n_1, \cdots, n_N\}$, from some obstacle to the point, where $n_1 \in \mathcal{O}$ and $n_N = n_a$. Now assume that at iteration k, the lattice point $n_k = \{p_{O_a}\}_k$ converges to its final value. The minimum-length path from n_1 to n_{k+1} is along the same path, $\{n_1, \cdots, n_{k+1}\}$. So, at iteration $k+1$, $n_{k+1} = \gamma n_k$.

Thus, n_{k+1}, converges to its final value in $k+1$ iterations. Since n_1 converges at time iteration 1, by induction n_N converges to its final value in N iterations. ■

Lemma 2 *The generalized distance-to-goal potential field, computed by equation 9, converges in a finite number of steps for all points, $n_{i,j}$, extended-lattice path-connected to the goal, n_G.*
Proof. Consider any such point, n_a. By assumption, there exists a minimum distance path, $p_{Ga} = \{n_1, \cdots, n_N\}$, from the goal to the point, where $n_1 = n_G$ and $n_N = n_a$. Now assume that at iteration k, the lattice point $n_k = \{p_{Ga}\}_k$ converges to its final value. The minimum-distance path from the goal to n_{k+1} is clearly $\{n_1, \cdots, n_{k+1}\}$. So, at iteration $k+1$, $n_{k+1} = \gamma + n_k$. Thus, n_{k+1}, converges to its final value in $k+1$ iterations. Since $n_1 = n_G$ converges at time iteration 1, by induction, n_N converges to its final value in N iterations. ■

Although these proofs assume certain initial conditions for $\Phi^B_{i,j}$ and $\Phi^D_{i,j}$, it is stated without proof that the field computations will converge to the same values for any initial conditions. Thus, if the obstacle pattern changes at time iteration k, $\Phi^B_{i,j}(k)$ and $\Phi^D_{i,j}(k)$ can serve as the new initial conditions. This facilitates tracking of moving obstacles.

Single Minimum — In previous works such as [10,13,16], the computation of the potential field consisted of three steps. First, an obstacle-repulsor field was computed. This field decayed from some maximum value at the surface of obstacles to zero far from obstacles. Second, a uniform, global basin of attraction with minimum value at the goal was constructed. Finally, the overall potential field was computed by superimposing these two fields. Such a scheme, however, often resulted in local minima of the field "uphill" from obstacles. Volpe and Khosla have proposed the use of superquadric functions in the construction of artificial potential fields to prevent the creation of such local minima [16]. However, they found that the technique did not solve the problem of local minima caused by obstacles containing large concavities pointing away from the goal.

Using the method proposed here, $\Phi^B_{i,j}$ serves the role of a obstacle-repulsor field. However, a global basin of attraction is not computed *independently* of the obstacle-repulsor field. Rather, the overall potential field, $\Phi^D_{i,j}$, contains *cumulative* information about $\Phi^B_{i,j}$ along the best path to the goal. This fact is essential in preventing undesirable local minima. Figure 5 shows a sample potential field calculated by directly adding the obstacle-repulsion field, $\Phi^B_{i,j}$, to a simple distance-to-goal field. Notice the local minimum occurring in the concavity "uphill" of the obstacle. However, as can be seen in figure 6, the potential field calculated using the cumulative $\Phi^B_{i,j}$ method has no such spurious local minimum. A proof of this property is given in lemma 3.

Lemma 3 *The lattice field, $\Phi^D_{i,j}$, calculated by 2-layer relaxation contains a unique local minimum at n_G over the set of lattice points which are extended-lattice path-connected to n_G.*
Proof. Clearly $\Phi^B_{i,j}(k) \geq 0 \ \forall k$. Therefore, by lemma 1, $\Phi^B_{i,j}$ converges to $\Phi^B_{i,j} > 0$. Now, for all

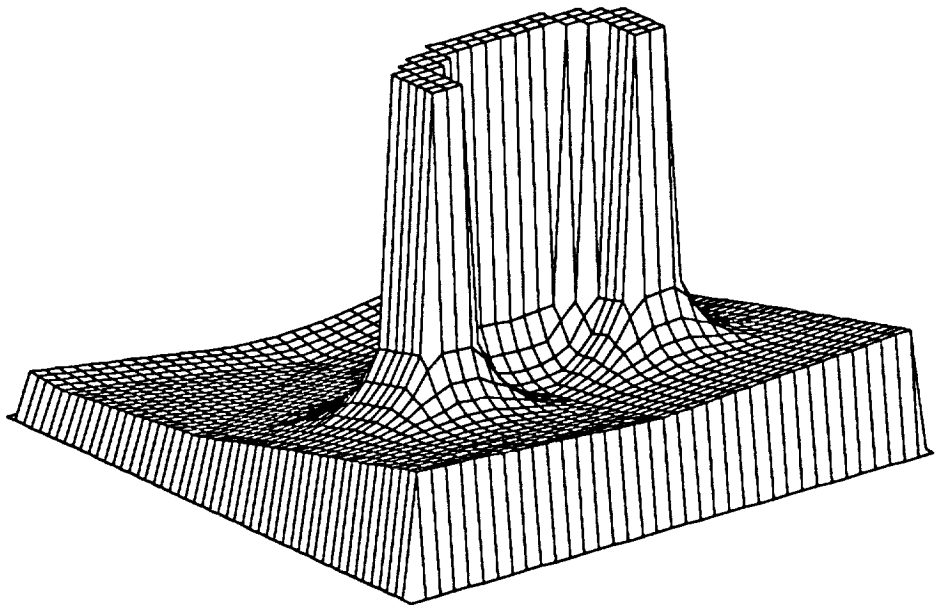

Figure 5: *Potential field computed by adding $\Phi^B_{i,j}$ to a distance-to-goal field has a local minimum "uphill" of the obstacle.*

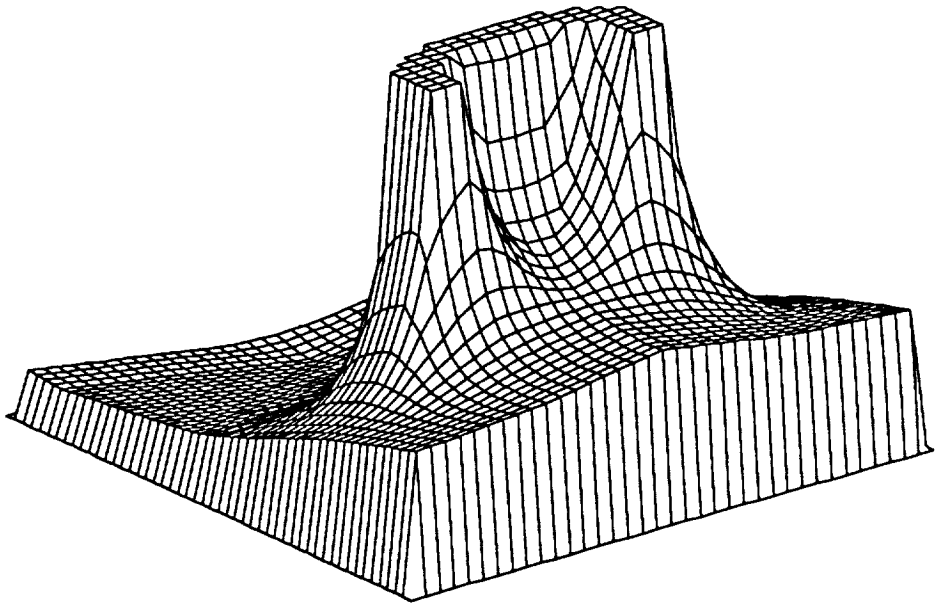

Figure 6: *Potential field using cumulative $\Phi^B_{i,j}$ has no local minima.*

points extended-lattice path-connected to goal, the calculation of $\Phi^D_{i,j}$ converges by lemma 2. Now, assume there exists a point, $n_a \neq n_G$, which is a lattice local minimum of $\Phi_{i,j}$. By assumption, $\Phi_{n_a} \leq \Phi_{n_b}\ \forall n_b \in N_8(n_a)$. However, since $n_a \neq n_G$, by equation 9 there exists $n_b \in N_8(n_a)$ such that $\Phi_{n_a} = \gamma + \Phi_{n_b} > \Phi_{n_b}$ which contradicts the assumption. ■

Potential Field Cross-Sectional Shape — The cross-sectional shape of the potential field is an important consideration. First consider the potential field which results from using the two layer relaxation method as presented. Consider an arbitrary point, n_a, in operational space and the associated optimal lattice-path, $p_{ga} = \{n_1, \cdots, n_N\}$, from the goal to n_a. Recall the relaxation equation for the overall potential from equation 9:

$$\Phi^D_{i,j}(k+1) = \Phi^B_{i,j} + \min_{n \in N_8(n_{i,j})} \{d_n + \Phi^D_n(k)\} \tag{10}$$

In steady state, at any point, $n_j = n_1 \cdots n_N$, along the path, p_{ga}, we can rewrite this relaxation equation as

$$\begin{array}{rcll} \Phi^D_{n_j} & = & \Phi^B_{n_j} + d_{n_j} + \Phi^D_{n_{j-1}} & \text{for } j > 1 \\ \Phi^D_{n_1} & = & 0 & \end{array} \tag{11}$$

where $d_{n_j} \in \{1, \sqrt{2}\}$ is the lattice distance between n_j and n_{j-1}. The solution to this equation is

$$\Phi^D_{n_a} = \sum_{j=2}^{N} \left(\Phi^B_{n_j} + d_{n_j}\right) \tag{12}$$

First consider the case where n_a is at the surface of an obstacle, i.e. $n_N \in \mathcal{O}$ and $n_{N-1} \notin \mathcal{O}$. Recall that $\Phi^B_{i,j} = \Phi^B_{max}\alpha^{k_o}$ where k_o is the distance to the nearest obstacle. If $\Phi^B_{max} \gg 1$, then $\Phi^B_{n_j} \gg d_{n_j}$ for n_j near n_a. This results in the geometric relation

$$\Phi^D_{n_j} = \frac{\Phi^B_{max}}{1-\alpha}\alpha^{N-j} + \nu \tag{13}$$

Now consider the case where n_a is far from any obstacle, thus, $\Phi^B_{n_j} \ll d_{n_j}$. This yields the linear relation

$$\Phi^D_{n_j} = k_g + \nu \tag{14}$$

where k_g is the distance from the goal to n_j along the optimal path, p_{gj}. In both cases, the constant ν incorporates the barrier potential accumulated in the vicinity of other obstacles.

At the expense of computational simplicity, the lattice relaxation method can be extended to obtain fields with other cross-sectional shapes. For example, we can modify the relaxation equation for $\Phi^D_{i,j}$ to

$$\Phi^D_{i,j}(k+1) = g\left(f\left(\Phi^B_{i,j}\right) + \min_{n \in N_8(n_{i,j})} \left\{d_n + g^{-1}\left(\Phi^D_n(k)\right)\right\}\right) \tag{15}$$

By specifying different functions, $f()$ and $g()$, the shape of the potential field can be altered. In order to retain a single local minimum, $g()$ must be positive and monotonically increasing. The function $f()$ need only be positive. The solution to equation 15 is

$$\Phi^D_{n_k} = g\left(\sum_{j=2}^{k}\left(f\left(\Phi^B_{n_j}\right) + d_{n_j}\right)\right) \tag{16}$$

3.4 Potential Field Interpolation

The potential field calculated above is defined only at lattice points (grid square centers). To obtain a continuous field, $\Phi^D_{x,y}$, it is necessary to interpolate between these points. Interpolation can be accomplished using the bilinear form:

$$\begin{aligned}
\Phi^D_{x,y} &= \Phi^D_{i,j}(1-\frac{\Delta x}{\delta})(1-\frac{\Delta y}{\delta}) + \Phi^D_{i+1,j}\frac{\Delta x}{\delta}(1-\frac{\Delta y}{\delta}) + \Phi^D_{i,j+1}(1-\frac{\Delta x}{\delta})\frac{\Delta y}{\delta} + \Phi^D_{i+1,j+1}\frac{\Delta x}{\delta}\frac{\Delta y}{\delta} \\
\Delta x &= x - \delta i \\
\Delta y &= y - \delta j \\
i &= \lfloor\frac{x}{\delta}\rfloor \\
j &= \lfloor\frac{y}{\delta}\rfloor
\end{aligned} \tag{17}$$

This formula represents nothing more than a weighted average of the field at the 4 nearest lattice points. The weights are dependent on the distances to the lattice points along the x and y axes.

The interpolation described in equation 17 can be implemented in hardware in the following manner: The output, $\Phi^D_{i,j}$, of each processor is connected to a common output summing node via a cascade of 2 attenuators. The 2 attenuation levels are set by *row* and *column* selector levels. For any given (x, y) access, 2 row selectors and 2 column selectors are set to attenuation values between 0 and 1, all other row and column selectors are set to 0. The interpolated field value is produced at the output of the summing node. A few properties of the interpolation formula and of the resulting continuous field, $\Phi^D_{x,y}$, follow.

Lemma 4 *If $\Phi^B_{i,j}$ is omitted in the calculation of $\Phi^D_{i,j}$, and there is a unique goal, n_G, then the lattice field, $\Phi^D_{i,j}$, does not contain any two points, n_a and n_b, both extended-lattice path-connected to n_G, such that $n_a \in N_8(n_b)$ and $\Phi^D_{n_a} = \Phi^D_{n_b}$.*

Proof. Assume that there are two such points. Then there must be two minimum distance paths, p_{Ga} and p_{Gb}, connecting each of the two points to the goal such that there are an equal number of diagonal path-segments and an equal number of non-diagonal path-segments in the two paths. A diagonal path segment can be replaced by two non-diagonal path segments without affecting the proof. Thus there must be a closed path, $\{p_{Ga}, p_{ab}, p_{bG}\}$ with an odd number of path segments. However it is impossible to have such a closed path with an *odd* number of segments. ■

Lemma 5 *If $\Phi^B_{i,j}$ is omitted in the calculation of $\Phi^D_{i,j}$, and there is a unique goal, n_G, the continuous field, $\Phi^D_{x,y}$ contains a unique local minimum, n_G.*

Proof. The interpolation function of equation 17, within the region, $0 < (x - i\delta) < \delta$ and $0 < (y - j\delta) < \delta$, is bilinear in x and y. Thus, minima must occur on the boundary of this region. Hence, local minima of $\Phi^D_{x,y}$ must occur on the line segments connecting neighboring lattice points (in the $N_4()$ sense). By lemma 4, neighboring lattice points cannot have equal values of $\Phi^D_{i,j}$, so minima must occur at lattice points themselves. By the piecewise bilinearity of $\Phi^D_{x,y}$, such a continuous local minima at a lattice point implies a lattice local minima at that point. However, by lemma 3, this can only occur at n_G. ■

If $\Phi^B_{i,j}$ is included in the calculation of $\Phi^D_{i,j}$, then lemma 5 can be modified to show that $\Phi^D_{x,y}$ has a unique local minimum except for the possible occurance of an equi-potential trough of length δ occurring between two neighboring lattice points having equal value. However, recall that the lattice potential field itself has a unique local minimum, and the length of the trough can be made arbitrarily small by increasing the resolution of the lattice.

4 Feedforward Neural-Controller

So far we have only concerned ourselves with various potential fields and have made no mention about how the fields are to be used to control the truck or similar autonomous vehicle. In this section we develop a feedforward neural network capable of using the potential field to control and navigate the truck to the goal.

4.1 Controller Structure

Previous work by Nguyen and Widrow [6,7] has shown that a feedforward neural network can successfully be trained to implement a nonlinear state feedback controller for backing a trailer-truck to a loading dock. The two segmented truck in consideration had the following state and control vectors (fig. 7).

$$\begin{aligned} \text{state} &= [x_{trailer}, y_{trailer}, \theta_{trailer}, \theta_{cab}] \\ \text{control} &= [\theta_{steering}] \end{aligned}$$

We refer to $\theta_{cab} - \theta_{trailer}$ as the jack angle, θ_{jack}. In the work of Nguyen and Widrow, the state vector (and 2 redundant inputs) were used as the controller input. This input vector was chosen to provide the controller with the necessary information to solve the backing problem. For the navigation problem, the previous choice of controller inputs is clearly not sufficient; information about obstacles must be provided to the controller. Attempts were made to use the previous controller input definition with a single, *fixed* obstacle and learn the obstacle position implicitly[2].

[2] F. Beaufays, personal communication, Stanford University, 1990

Although this scheme was able to steer the truck clear of an obstacle, extension to multiple obstacles was not feasible. Furthermore, the method required retraining for each new obstacle position.

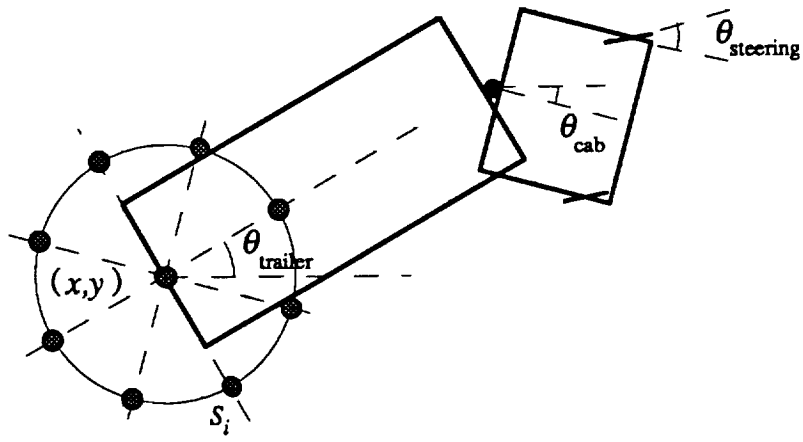

Figure 7: *Truck state variables and field interpolation points.*

In developing an alternate set of controller inputs, a set of field interpolation points, $\{s_i\}$, configured in a circle about the rear of the truck was defined in the local coordinate system of the truck (fig. 7). During each discrete control cycle, the local coordinates of the set of points, $\{s_i\}$, were converted into global coordinates by the appropriate transformation. The field value, $\Phi^D_{x,y}$, was measured at these locations giving the set $\{\Phi^D_{s_i}\}$. The difference between the field level at each of the outer points and the level at the center point (fig. 7), was computed:

$$\Delta_i = \Phi^D_{s_i} - \Phi^D_{s_{center}} \qquad \text{for } i = 1 \cdots N \tag{18}$$

The vector

$$\Delta \mathbf{s} = [\Delta_1, \cdots, \Delta_N] \tag{19}$$

gave a measure of the directional derivative of the potential field in N directions. This vector, combined with θ_{jack} and appropriately scaled, constituted the input vector to a 2-layer, sigmoidal feedforward network with 20 hidden units (fig. 8). The overall navigation–control structure, including this feedforward network and the 2-layer lattice network described previously, is shown in figure 9.

4.2 Controller Training

Training of the feedforward neural-controller was performed using backpropagation-through-time in an obstacle-free operational space. First, $\Phi^D_{i,j}$ was computed for this space. Note that, since there were no obstacles, $\Phi^B_{i,j} = 0$ and $\Phi^D_{i,j}$ consisted simply of the distance-to-goal measure. The truck was then placed in random initial configurations within the space and trained to back to the goal, while minimizing θ_{jack}. Training was carried out as in the work of Nguyen and Widrow [7]

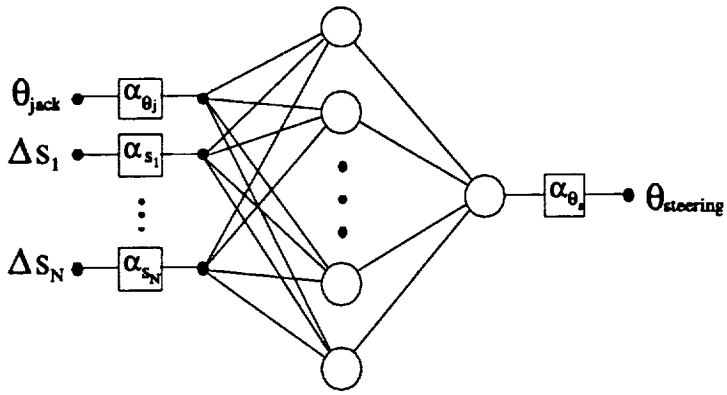

Figure 8: *Structure of the neural-controller consisting of feedforward network and scaling factors.*

except that the docking-angle constraint was lifted by setting

$$[\epsilon_x, \epsilon_y, \epsilon_{\theta_t}, \epsilon_{\theta_c}] = [x_{dock} - x_{final}, y_{dock} - y_{final}, 0, 0] + \theta_{jack}[0, 0, 1, -1] \tag{20}$$

Imposing the first term caused the truck to dock with the goal while imposing the second term caused the controller to minimize the angle between the cab and trailer, thus preventing jack-knifing. Training required about 1000 complete docking cycles, with initial conditions being chosen progressively further from goal and at larger offset angles. When the truck was then placed in an operational space with obstacles, the obstacles warped $\Phi^D_{i,j}$. The truck was now able to navigate around obstacles without any further training. The truck trajectories from different initial conditions in the sample obstacle space are shown in figures 10 and 11.

Consider what task-knowledge was gained by the feedforward network during the training process which permitted it to successfully use $\Phi^D_{x,y}$ to navigate. If the controller had been provided with only truck state information, the controller would have learned the task of directly driving the state vector to zero, knowledge which would not have permitted obstacle avoidance. However, the particular choice of controller inputs, $[\theta_{jack}, \Delta\mathbf{s}]$, led to different task-knowledge. Previously we mentioned that the desired truck behavior was to follow the negative gradient of the potential field, $-\nabla\Phi^D_{x,y}$. In figure 10, the desired behavior is evidenced. Enough information is provided in $\Delta\mathbf{s}$ to accomplish the navigational task.

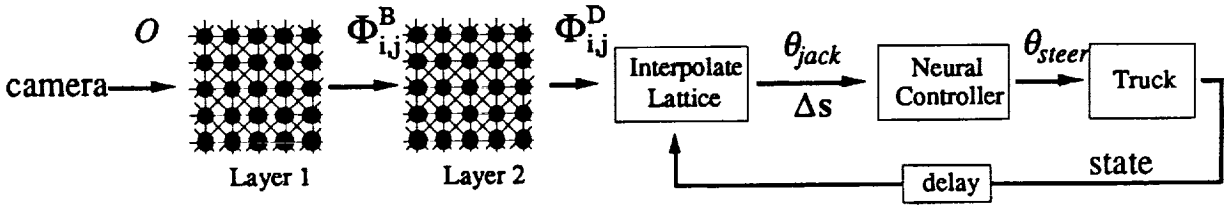

Figure 9: *Overall navigation/control structure comprised of a two-layered systolic array, a lattice interpolation function, and a feedforward network controlling truck steering.*

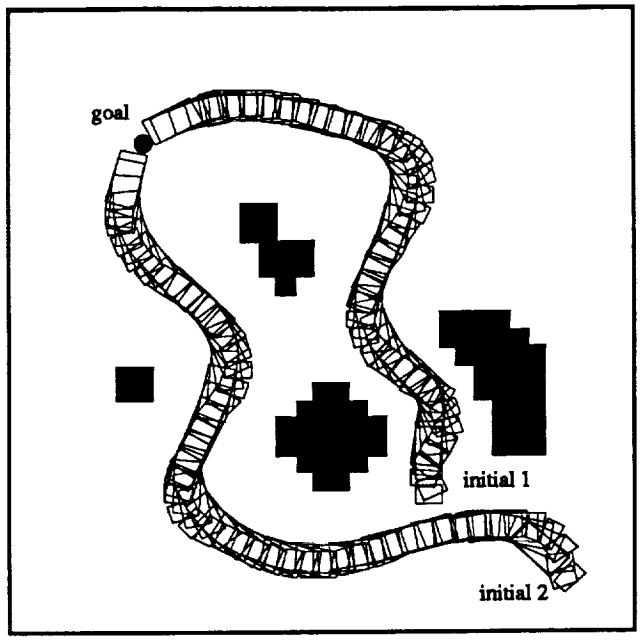

Figure 10: *Truck trajectories from two initial conditions in the sample obstacle space.*

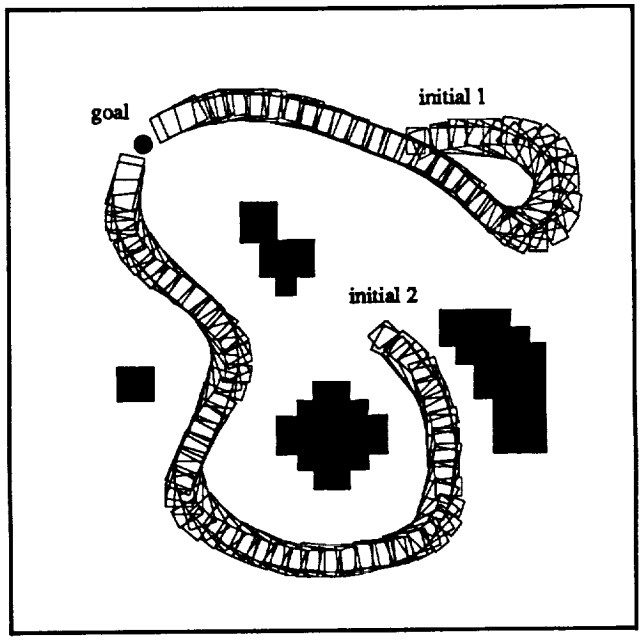

Figure 11: *Trajectories showing desired behavior of driving "uphill" when obstacle configuration prevents driving "downhill."*

However, the truck dynamics do not permit motion in arbitrary directions. Hence, simply following $-\nabla\Phi^D_{x,y}$ is not always feasible, nor necessarily desirable. In figure 11, initial condition 2, the truck is started in a narrow corridor facing the wrong way. The truck backs *uphill* first before circling around the obstacle. This is correct since attempting to turn around sharply, trying to immediately follow $-\nabla\Phi^D_{x,y}$, would result in a collision.

In reality, the feedforward network combined the objective of following $-\nabla\Phi^D_{x,y}$ with the dynamic constraints of the truck, thus attempting to minimize the directional derivative of $\Phi^D_{x,y}$ within an arc of feasible directions. This was done while minimizing θ_{jack}. It is interesting to observe that this complex behavior was learned in an obstacle-free space.

5 Conclusion

A hybrid-network method for obstacle avoidance in the truck-backing system of Nguyen and Widrow has been presented. The method uses a relatively simple, low-level algorithm easily implemented in hardware or software.

A key advantage of the technique is the separation of the systolic array structure from the feedforward network. The systolic array computations are independent of the truck dynamics, constraints, etc., and serve simply to distribute information about goal and obstacle locations. These computations, carried out in parallel, are quick. In the examples given, the systolic array converged in less than 40 cycles. It is important not to view this procedure as path-planning. Path-planning requires consideration of both obstacle locations and truck configuration. There is actually no path-planning being done, as the computations serve only as a mechanism for distributing information. The feedforward neural network, on the other hand, is trained independently of any particular obstacle configuration and thus can be held fixed after initial training. No additional training is needed when a new obstacle pattern is presented.

Currently, grid-squares are either impenetrable obstacles or completely free. The technique presented can easily be extended to handle variable terrain difficulties by making Φ^B_{max} and Φ^D_{max} vary for each obstacle square. This could be used, for example, to avoid rocky terrain if a clear path was available, yet allow the rocky terrain to be crossed if it constituted the only path. Specific choices of Φ^B_{max} depend on the trading-off of terrain difficulty and path length.

In situations where tight maneuvering is necessary to negotiate obstacles, the method presented may fail. It would, in those cases, be desirable to further train the feedforward network in the presence of obstacles to tune the control. This might also require additional sensory input to the controller. One may also observe that the method could generate a path which passes through a barrier aperture too narrow for the truck to fit through. It is envisioned that the truck controller could determine this while navigating. When such an aperture is encountered, the controller could modify the obstacle space description by increasing the local obstacle level (the input to the 1^{st} layer of the systolic array) until the potential field shifted to favor a different direction. This could be accomplished in real-time due to the speed of convergence of the array computations.

The systolic array requires obstacle information for the entire space in which navigation is to occur. In the case where the vehicle must navigate over a long distance for which the entire obstacle space is not known, the method could be used to perform local path-planning assuming that the general desired direction of travel is known.

Currently, work is underway to extend the technique presented to include a docking angle constraint and eventually to maneuver multiple linked objects, such as a robotic arm. This would involve "path-planning" for multiple points on the object. Work is also being done to characterize the warping of $\Phi^p_{i,j}$ resulting from multiple minimization criteria.

References

[1] F. Pineda. Generalization of backpropagation to recurrent neural networks. *Physical Review Letters*, 18(59):2229–2232, 1987.

[2] D. Rummelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rummelhart and J. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8. The MIT Press, Cambridge, MA, 1986.

[3] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, August 1974.

[4] M. Jordan. Generic constraints on underspecified target trajectories. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 217–225, Washington, DC, June 1989.

[5] K. Narendra and K. Parthasarathy. Identification and control of dynamic systems using neural networks. *IEEE Transactions on Neural Networks*, pages 4–27, March 1990.

[6] D. Nguyen and B. Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 357–363, Washington, DC, June 1989.

[7] D. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10(3):18–23, April 1990.

[8] D. Psaltis, A. Sideris, and A. Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, 8(2):17–21, April 1988.

[9] R. Brooks. Solving the find-path problem by good representation of free space. In *Proceedings 2nd American Association of Artificial Intelligence Conference*, pages 381–386, 1982.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[11] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[12] S. Udupa. Collision detection and avoidance in computer controlled manipulators. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977.

[13] B. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proceedings of SME Conference on Robotics Research: The Next Five Years and Beyond*, Bethlehem, PA, August 1984.

[14] W. Newman and N. Hogan. High speed robot control and obstacle avoidance using dynamic potential functions. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 1, pages 14–24, 1987.

[15] C. Warren. Global path planning using artificial potential fields. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 1, pages 316–321, 1989.

[16] R. Volpe and P. Khosla. Manipulator control with superquadric artifical potential functions: theory and experiments. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 1423–1436, December 1990.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-177575		

4. Title and Subtitle
Cascading a Systolic Array and a Feedforward Neural Network for Navigation and Obstacle Avoidance Using Potential Fields

5. Report Date
February 1991

6. Performing Organization Code

7. Author(s)
Edward S. Plumer

8. Performing Organization Report No.
A-91066

10. Work Unit No.

9. Performing Organization Name and Address
Department of Electrical Engineering
Stanford University
Stanford, CA 94305-4055

11. Contract or Grant No.
NGT-50642

13. Type of Report and Period Covered
Contractor Report

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes
Point of Contact: C. Jorgensen, Ames Research Center, MS 244-4, Moffett Field, CA 94035-1000
(415) 604-6725 or FTS 464-6725

16. Abstract

The goal of this research is the development of a technique for vehicle navigation and control in the presence of obstacles. A potential function has been devised that peaks at the surface of obstacles and has its minimum at the proper vehicle destination. This function is computed using a systolic array and is guaranteed not to have local minima. A feedforward neural network is then used to control the steering of the vehicle using local potential field information. In this case, the vehicle is a trailer-truck backing up. Previous work has demonstrated the capability of a neural network to control steering of such a trailer-truck backing to a loading platform, but without obstacles. Now, the neural network has been able to learn to navigate a trailer-truck around obstacles while backing toward its destination. The network is trained in an obstacle free space to follow the negative gradient of the field, after which the network is able to control and navigate the truck to its target destination in a space of obstacles which may be stationary or movable.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Obstacle avoidance, Navigation, Potential field, Neural network, Systolic array	Unclassified-Unlimited Subject Category - 63

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	24	A02